Reihe: Telekommunikation @ Mediendienste · Band 14

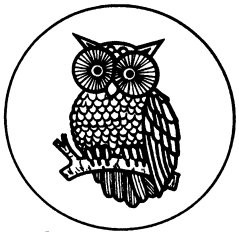
Herausgegeben von Prof. Dr. Dr. h. c. Norbert Szyperski, Köln, Prof. Dr. Udo Winand, Kassel, Prof. Dr. Dietrich Seibt, Köln, Prof. Dr. Rainer Kuhlen, Konstanz, Dr. Rudolf Pospischil, Brüssel, Prof. Dr. Claudia Löbbecke, Köln, und Prof. Dr. Christoph Zacharias, Köln

PD Dr.-Ing. habil. Martin Engelien
Dipl.-Inf. Jens Homann (Hrsg.)

Virtuelle Organisation und Neue Medien 2002

Workshop GeNeMe2002
Gemeinschaften in Neuen Medien

TU Dresden, 26. und 27. September 2002



JOSEF EUL VERLAG
Lohmar · Köln

Die Deutsche Bibliothek – CIP-Einheitsaufnahme

Virtuelle Organisation und Neue Medien 2002 / Workshop GeNeMe 2002 – Gemeinschaften in Neuen Medien – TU Dresden, 26. und 27. September 2002. Hrsg.: Martin Engelien ; Jens Homann. – Lohmar ; Köln : Eul, 2002
(Reihe: Telekommunikation und Medienwirtschaft ; Bd. 14)
ISBN 3-89936-007-9

Josef Eul Verlag GmbH
Brandsberg 6
53797 Lohmar
Tel.: 0 22 05 / 90 10 6-6
Fax: 0 22 05 / 90 10 6-88
http://www.eul-verlag.de
info@eul-verlag.de

Printed in Germany
Druck: RSP Köln

Bei der Herstellung unserer Bücher möchten wir die Umwelt schonen. Dieses Buch ist daher auf säurefreiem, 100% chlorfrei gebleichtem, alterungsbeständigem Papier nach DIN 6738 gedruckt.

Technische Universität Dresden
Fakultät Informatik • Institut für Angewandte Informatik
Privat-Dozentur Angewandte Informatik

PD Dr.-Ing. habil. Martin Engelien

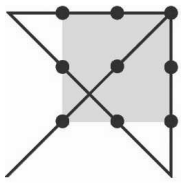
Dipl.-Inf. Jens Homann

(Hrsg.)

an der
Fakultät Informatik der Technischen Universität Dresden

in Zusammenarbeit mit der
Gesellschaft für Informatik e.V.,
GI-Regionalgruppe Dresden

gefördert von der Klaus Tschira Stiftung
gemeinnützige Gesellschaft mit beschränkter Haftung



am 26. und 27. September 2002
in Dresden
http://pdai.inf.tu-dresden.de/geneme
Kontakt: Thomas Müller (geneme@pdai.inf.tu-dresden.de)

D.2. Wege zu einer Software-Komponenten-Industrie - Erfolgsfaktoren für die Bildung von virtuellen Gemeinschaften in der Software-Entwicklung

Oliver Höß[1]

Anette Weisbecker[2]

Fraunhofer-Institut für Arbeitswirtschaft und Organisation

Universtität Stuttgart

1. Einleitung

Software hat sich in den letzten Jahren zu einem der zentralen Faktoren innerhalb der industriellen Wertschöpfungskette entwickelt. Die Wertschöpfung der Primärbranche, d.h. der softwareentwickelnden Unternehmen in Deutschland, übersteigt mit ca. 25 Mrd. Euro (im Jahr 2000) die Wertschöpfung im Sektor Landwirtschaft, Forstwirtschaft und Fischerei um ca. 20 % [GfK+2000]. Außerdem hängt die Produktivität fast aller produzierenden Branchen sowie des Dienstleistungssektors weitestgehend von der Unterstützung durch geeignete Software ab. Aus dieser großen Bedeutung ergibt sich ein enormes Einsparpotenzial, das durch die effiziente Erstellung und durch den effizienten Einsatz von Software realisiert werden kann.

Software wird dabei auf vielfältige Art und Weise eingesetzt (vgl. auch [Weis2000, HöWe2001]): zur Unterstützung der internen Geschäftsprozesse, zur Unterstützung von Dienstleistungen für Kunden (B2C), zur Unterstützung der Zusammenarbeit mit Geschäftspartnern (B2B) sowie als Software in oder als Bestandteil von Produkten (Embedded Software).

Die in diesen Bereichen entstehenden Lösungen besitzen heutzutage meist eine Komplexität, die eine Erstellung der Software innerhalb des Zeit- und Kostenrahmens zu einer anspruchsvollen Aufgabe machen. Die komponentenbasierte Software-

[1] Dipl.-Inf. Oliver Höß ist wissenschaftlicher Mitarbeiter und Projektleiter im Competence Center Software-Management am Fraunhofer-Institut für Arbeitswirtschaft und Organisation (IAO), Nobelstr. 12, 70569 Stuttgart, Tel. +49 (711) 970 – 2409, E-Mail: Oliver.Hoess@iao.fhg.de

[2] Dr.-Ing. habil. Dipl.-Inform. Anette Weisbecker ist Institutsdirektorin des Fraunhofer IAO sowie Leiterin des Competence Centers Software-Management, Tel. +49 (711) 970 – 2400, E-Mail: Anette.Weisbecker@iao.fhg.de

entwicklung hat sich als ein geeignetes Paradigma erwiesen, um diese Anforderungen zu bewältigen. Insbesondere im E-Business-Bereich wird dieser Ansatz erfolgreich eingesetzt (siehe auch [HöWe2002b]).

Der Grundgedanke der komponentenbasierten Softwareentwicklung ist es, Systeme nicht immer von Grund auf neu zu erstellen, sondern bereits vorgefertigte qualitativ hochwertige Bausteine (sog. Komponenten) nach dem Baukasten-Prinzip zusammenzusetzen. Dabei sind Komponenten technisch und funktional abgeschlossene Einheiten, die über genau spezifizierte Schnittstellen miteinander zusammenarbeiten. Auf diese Art und Weise kann die Komplexität der zu erstellenden Systeme reduziert werden. Qualitativ hochwertige Systeme können somit in relativ kurzer Zeit erstellt werden.

Desweiteren bildet die klare Strukturierung von Software in einzelne Komponenten die Grundlage für die Etablierung einer funktionierenden „Software-Komponenten-Industrie“, in der Komponenten-Hersteller und System-Integratoren virtuelle Gemeinschaften bilden, um kooperativ Endprodukte in Form von komplexen Software-Systemen zu erstellen.

Der Faktor der Wiederverwendung spielt in der komponentenbasierten Software-Entwicklung die entscheidende Rolle: nur wenn ein genügend großer Vorrat an wiederverwendbaren Komponenten besteht, auf die der jeweilige Verwender auch zugreifen kann, können durch Wiederverwendung dieser bestehenden Komponenten die Vorteile des Komponentenansatzes genutzt werden.

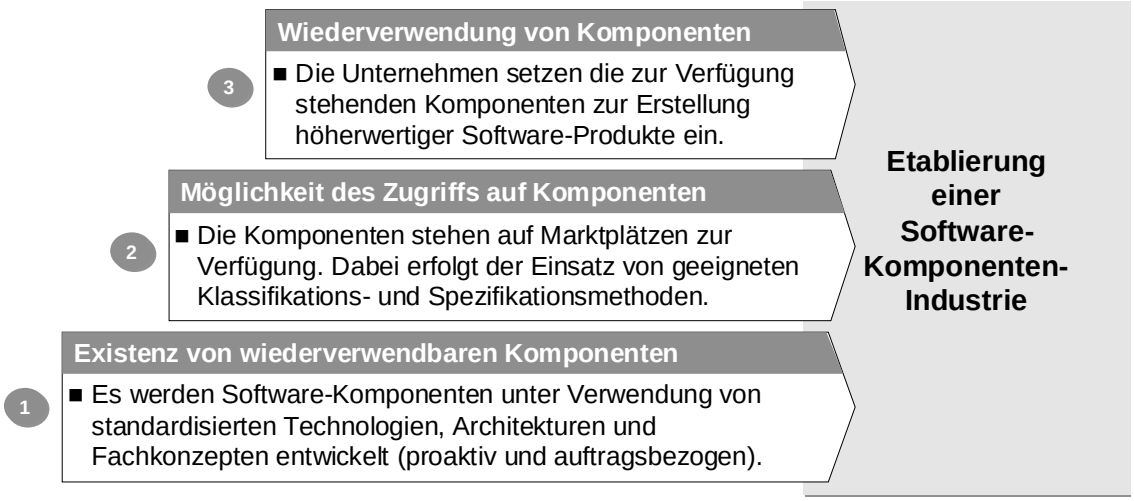


Abb. 1: Schritte zu einer Software-Komponenten-Industrie

In Abb. 1 sind die drei wesentlichen Schritte dargestellt, die für die Etablierung einer Software-Komponenten-Industrie notwendig sind:

- Die Existenz von wiederverwendbaren Software-Komponenten: Es müssen zur Wiederverwendung geeignete Komponenten zur Verfügung stehen. Dies ist nur möglich, wenn standardisierte Technologien und Software-Architekturen eingesetzt werden. Neben diesen eher technischen Aspekten müssen auch die fachlichen Konzepte der einzelnen Komponenten miteinander „kompatibel“ sein. Die Entwicklung dieser Komponenten erfolgt dabei sowohl proaktiv, d.h. ohne einen konkreten Auftrag als auch auftragsbezogen.

- Möglichkeit des Zugriffs auf die Komponenten: Auf den Vorrat an wiederverwendbaren Komponenten muss ein Zugriff möglich sein, d.h. ein potenzieller Verwender der Komponente muss über geeignete Mechanismen nach den gesuchten Komponenten recherchieren können. Dazu bieten sich Markplätze im Internet an, die die Komponenten nach standardisierten Klassifikationskriterien geordnet zur Verfügung stellen und die Funktionalität der Komponenten unter Verwendung einer standardisierten Spezifikationsmethode beschreiben.

- Wiederverwendung von Komponenten: Wenn geeignete Komponenten existieren und diese auch zugreifbar sind, sollten diese auch durch die Unternehmen wiederverwendet werden. Dieser eigentlich selbstverständliche Schritt wird in der Praxis oft nicht durchgeführt. Gründe dafür werden in Kap. 2 beschrieben.

In anderen, schon länger bestehenden und somit reiferen Industriezweigen, wie z.B. der Automobil- oder Elektronikindustrie sind Zulieferbeziehungen, oft auch in mehrstufiger Form, schon seit längerem realisiert. Auf diese Art und Weise kann sich jedes Unternehmen auf seine eigentlichen Kernkompetenzen, d.h. entweder auf die Herstellung von einzelnen Komponenten oder die Integration zum Gesamtsystem, konzentrieren und somit die jeweilige Aufgabe mit größerer Effizienz bewältigen.

Durch die Immaterialität von Software ist der Bereich der Softwareentwicklung eigentlich besonders dazu geeignet, derartige Gemeinschaften hervorzubringen, da der weltweite Transport von Software in allen Entwicklungsstadien, d.h. sowohl während des Entwurfs als auch als Produkt, über das Internet weder eine nennenswerte Zeit benötigt noch nennenswerte Kosten verursacht.

Untersuchungen (z.B. [DiEs2001, GfK+2000]) und auch die Erfahrung des Fraunhofer IAO bei der Arbeit mit Industriepartnern zeigen jedoch, dass eine derartige Industrie nicht bzw. nur in Ansätzen existiert.

In diesem Beitrag werden nun Faktoren vorstellt, die die Bildung einer „Software-Komponenten-Industrie“ hemmen. Ausgehend von diesen Hemmnissen werden dann Schritte aufgezeigt, die notwendig sind, um dieses Ziel, eine Software-Komponenten-Industrie zu etablieren, zu erreichen.

2. Hemmnisse der Bildung einer Software-Komponenten-Industrie

Durch die Entwicklung von standardisierten Komponententechnologien, wie z.B. die Technologien der J2EE-Architektur von Sun oder die Technologie in Microsoft’s .NET-Framework, existiert die technologische Grundlage, um wiederverwendbare Software-Komponenten zu entwickeln. Die Tatsache dieser technologischen Standardisierung hat die Bildung von Komponentenmärkten im Internet ermöglicht. Beispiele hierfür sind die Marktplätze ComponentSource[3], Flashline[4], CompoNex[5] oder die Open-Source Entwicklungsplattform SourceForge[6]. Trotz dieser begünstigenden Faktoren hat sich jedoch bisher keine funktionierende Software-Komponenten-Industrie entwickelt (siehe [DiEs2001]), obwohl in vielen Branchen durchaus der Wunsch besteht, die Fertigungstiefe in der Softwareentwicklung zu reduzieren und mehr auf extern gefertigte Komponenten zurückzugreifen ([GfK+2000], Seite 6).

Diese Tatsache ist durch eine Reihe von Hemmnissen begründet, die die Bildung einer Software-Komponenten-Industrie behindern. Die wesentlichen Hemmnisse werden in den folgenden Abschnitten erläutert. Ein Überblick ist in Abb. 2 dargestellt.

[3] www.componentsource.de

[4] www.flashline.com

[5] www.componex.biz

[6] www.sourceforge.net

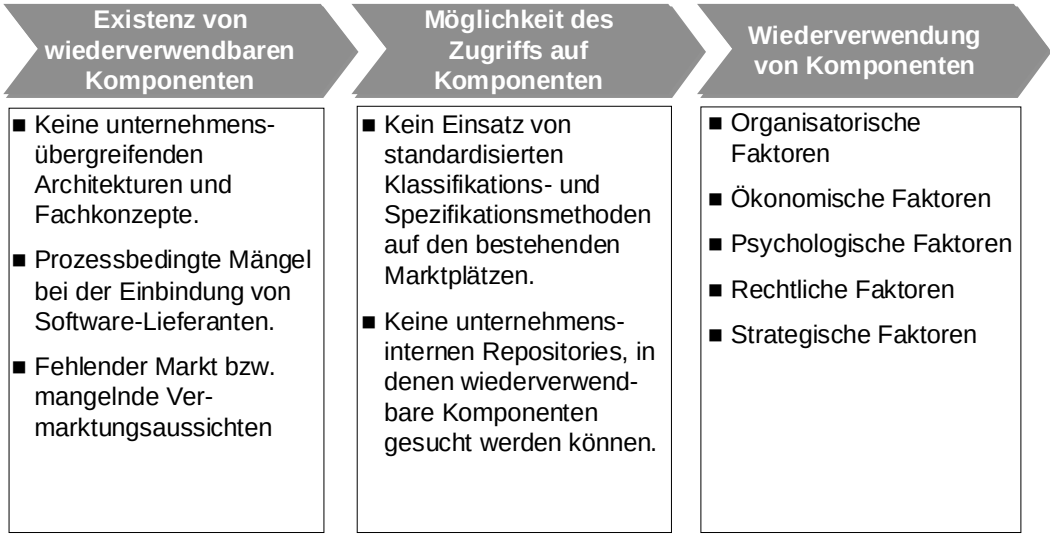


Abb. 2: Hemmnisse der Bildung einer Software-Komponenten-Industrie

2.1 Hemmnisse für die Existenz von wiederverwendbaren Komponenten

Ein wesentlicher Faktor, der die Entwicklung von wiederverwendbaren Software-Komponenten behindert, ist der Mangel an unternehmensübergreifenden Architekturen und Fachkonzepten, d.h. Konzepten, die auf den bestehenden Komponenten-Technologien aufbauen und die Entwicklung von unternehmensübergreifend einsetzbaren Komponenten für die Konstruktion von komplexeren Anwendungssystemen erst ermöglichen. In vielen, insbesondere auch größeren Unternehmen, tritt dieser Effekt auch innerhalb eines Unternehmens auf, d.h. innerhalb desselben Unternehmens existieren eine Reihe von unterschiedlichen Architekturen und Fachkonzepten, die auch eine Wiederverwendung innerhalb dieses Unternehmens erschweren oder gar unmöglich machen.

Im Falle der Vergabe von Entwicklungsaufträgen an externe Software-Lieferanten tritt ein ähnlicher Effekt auf. In vielen Fällen werden den externen Zulieferern keine genauen Vorgaben hinsichtlich der Software-Architektur gemacht, so dass bei Unternehmen, die viele Entwicklungsaufträge an externe Partner vergeben, oft eine heterogene Systemlandschaft aus Insellösungen entsteht.

Für viele kleinere Firmen, die eigentlich als Komponenten-Lieferanten in einem Markt auftreten könnten, sind der bisher nicht existierende Markt und die dadurch resultierenden mangelhaften Vermarktungsaussichten eine Barriere, die sie daran

hindert, den Entwicklungsaufwand zu investieren, der für eine initiale Entwicklung von wiederverwendbaren Komponenten notwendig ist.

2.2 Hemmnisse beim Zugriff auf Software-Komponenten

Wenn man davon ausgeht, dass geeignete Software-Komponenten für einen bestimmten Einsatzzweck existieren, ist es in vielen Fällen nicht möglich, diese zu verwenden, weil die Kenntnis der Existenz dieser Komponenten bei potenziellen Verwendern nicht vorhanden ist.

Die bereits bestehenden Komponenten-Marktplätze, wie z.B. Flashline oder ComponentSource, benutzen unterschiedliche Klassifikationsschemata, nach denen die Komponenten abgelegt sind, was eine Suche für den Anwender erschwert. Auch die Spezifikation bzw. Dokumentation der einzelnen Komponenten ist sehr unterschiedlich, so dass der Einsatz bzw. die Prüfung eines möglichen Einsatzes einer Komponente in vielen Fällen unnötig schwierig ist.

Diese mangelnde Möglichkeit, vorhandene Komponenten zu finden, setzt sich auch auf der unternehmensinternen Ebene fort. In den wenigsten Unternehmen ist ein sog. Repository vorhanden, in dem die bereits im Unternehmen entwickelten Software-Komponenten verzeichnet sind, so dass eine Wiederverwendung i.d.R. nicht oder nur zufällig erfolgt.

Die Effizienz des Suchprozesses ist von entscheidender Bedeutung. Insbesondere darf der Suchvorgang nicht länger dauern, als eine Neuimplementierung einer Komponente.

2.3 Hemmnisse bei der Wiederverwendung von Software-Komponenten

Selbst unter der Voraussetzung, dass zur Wiederverwendung geeignete Komponenten existieren und entsprechende Mechanismen vorhanden sind, um diese zu finden, bestehen durchaus noch weitere Hemmnisse, die die Wiederverwendung von Software-Komponenten behindern.

Zum einen sind dies organisatorische Faktoren. In den meisten Unternehmen existiert keine organisatorische Einheit, die für die Wiederverwendung von extern oder intern entwickelten Software-Komponenten zuständig ist.

Durch die mangelnde Ermittlung von Kennzahlen und dem fehlenden Controlling von Entwicklungsprojekten ist dem Management oder der Projektleitung in vielen Fällen nicht bewusst, welche ökonomischen Vorteile die Wiederverwendung von bestehenden Komponenten besitzt.

Auch psychologische Faktoren spielen eine Rolle. Viele Entwickler besitzen einen gewissen „Forschertrieb“ und verwenden nur ungern bereits bestehende Software. Da sie in vielen Fällen keinen Anreiz und keine Zeit haben, vor jeder Entwicklung eine Marktrecherche nach eventuell verwendbaren Komponenten durchzuführen, unterbleibt dies in vielen Fällen.

Die ungeklärte rechtliche Situation, z.B. die Haftungsfrage bei einem Fehler des Gesamtsystems, der durch einen Fehler einer Teilkomponente hervorgerufen wird, führt dazu, dass Unternehmen teilweise Bedenken haben, Fremdkomponenten in ihre Systeme zu integrieren. Diese Bedenken können auch durch strategische Überlegungen hervorgerufen werden. Beispielsweise ist es in vielen Fällen strategisch unerwünscht, sich auf Teilkomponenten eines kleinen Herstellers zu verlassen, dessen wirtschaftliche Zukunft nicht gesichert ist. Die Entwicklung der beiden vergangenen Jahre, in denen viele kleinere Firmen ihre Geschäftstätigkeit aufgeben mussten, hat gezeigt, dass diese Bedenken berechtigt sind.

In Fällen, in denen Software das wesentliche Alleinstellungsmerkmal ist, wie z.B. bei Online-Finanzdienstleistern, ist dies ebenfalls oft ein Grund, an dieser Stelle keine extern bezogenen Komponenten zu verwenden, um dieses Alleinstellungsmerkmal gegenüber den Mitbewerbern nicht zu verlieren.

3. Erfolgsfaktoren für den Weg zu einer Software-Komponenten-Industrie

Um trotz der in Abschnitt 2 genannten Hemmnissen das Paradigma einer Software-Industrie umsetzen zu können, müssen die dafür notwendigen Voraussetzungen geschaffen und durch geeignete Maßnahmen unterstützt werden. In diesem Abschnitt werden dazu eine Reihe von Lösungsansätzen vorgestellt, die einzelne Teilprobleme adressieren und somit wichtige Erfolgsfaktoren für die Etablierung einer Software-Komponenten-Industrie darstellen:

- Entwicklung von Prozess- und Vorgehensmodellen für die kollaborative Software-Entwicklung

- Etablierung von unternehmensübergreifenden Komponenten-Marktplätzen und unternehmensinternen Komponenten-Repositories
- Entwicklung eines standardisierten Klassifikationsschemas für Software-Komponenten
- Entwicklung einer standardisierten Spezifikationsmethodik für Software-Komponenten
- Entwicklung von unternehmensübergreifenden Fachkonzepten
- Einführung der Wiederverwendung als Teil der Unternehmensstrategie
- Klärung der rechtlichen Grundlagen

Diese Schritte werden in den folgenden Abschnitten näher erläutert. Sie sind jedoch nicht als vollständige Aufzählung zu betrachten, sondern als Nennung der wesentlichen Schritte.

3.1 Entwicklung von Prozess- und Vorgehensmodellen für die kollaborative Software-Entwicklung

Ein wesentlicher Faktor für die Etablierung einer Software-Komponenten-Industrie ist die Entwicklung von geeigneten Vorgehensmodellen. Mit dem Rational Unified Process [Kruc1999], Catalysis [DsWi1998], Select Perspective [AlFr1998] sowie dem überarbeiteten V-Modell [DrWi1999] existiert zwar bereits eine Reihe von Vorgehensmodellen für die komponentenbasierte Softwareentwicklung, sie besitzen jedoch eindeutig noch einige Defizite, insbesondere im Bereich der Wiederverwendung von bereits bestehenden Komponenten (siehe [FeInLo2002]).

Diese Defizite werden in Abb. 3 verdeutlicht: Alle Vorgehensmodelle unterstützen die Entwicklung von Komponenten (Design for Component). Die Entwicklung von Anwendungssystemen aus Komponenten (Design from Component) wird nicht durch alle Vorgehensmodelle unterstützt. Kaum unterstützt wird der in der Praxis äußerst wichtige Fall der Transformation von bestehenden Systemen in komponentenbasierte Systeme (Design to Component). Die Vorgehensweisen Design for Component, Design from Component sowie Design to Component wurden im Rahmen des Verbundforschungsprojekts KoSPuD (Komponentenbasierte Software für Produkte und Dienstleistungen) entwickelt [Weis2000].

Die Wiederverwendung von Komponenten wird zwar in allen Vorgehensmodellen angesprochen, jedoch nur äußerst unzureichend durch konkrete Vorschläge und Handlungsanweisungen unterstützt. Die in einer Komponenten-Industrie sehr wichtige Tätigkeit der Komponenten-Evaluation und –Beschaffung wird nur durch das V-Modell konkret berücksichtigt (siehe auch [FeInLo2002]).

	RUP	**Catalysis**	**Perspective**	**V-Modell**
Modellierungsmethode	UML	UML	UML	Methodenunabhängig (UML, SA, SD, ...)
Werkzeugunterstützung	Rational Suite	Produkt-unabhängig	Select Component Factory	Produkt-unabhängig
Design for Component	√	√	√	√
Design from Component	bedingt	bedingt	√	√
Design to Component	-	bedingt	-	-
Wiederverwendung	bedingt	bedingt	bedingt	bedingt
Komponenten-Evaluation und -Beschaffung	-	-	-	√

Abb. 3: Komponentenbasierte Vorgehensmodelle im Vergleich

Das Ziel muss es also sein, Vorgehensmodelle zu entwickeln, bei denen statt der Eigenentwicklung von Systemen die Konstruktion von Systemen aus vorgefertigten Bausteinen im Mittelpunkt steht. Dabei nehmen Tätigkeiten wie die Evalution, Auswahl und Beschaffung von Komponenten einen hohen Stellenwert ein.

Ein weiterer wichtiger Bestandteil von derartigen Vorgehensmodellen sollten Leitlinien für das Outsourcing von Entwicklungsleistungen sein. In vielen Fällen werden keine für den jeweiligen Anwendungsfall geeigneten Komponenten existieren, so dass die Entwicklung an externe Zulieferer vergeben werden muss. Für diesen Fall sind insbesondere Leitlinien vorzugeben, wie eine unternehmensübergreifende Qualitätssicherung sowie eine Sicherstellung der Konformität mit der durch den Auftraggeber vorgegebenen Spezifikation und Architektur vorzunehmen ist.

3.2 Etablierung von Komponenten-Marktplätzen und unternehmensinternen Repositories

Um sicherzustellen, dass bestehende Komponenten zum Zweck der Wiederverwendung auch gefunden werden können, müssen Komponenten-Marktplätze etabliert werden, auf denen diese zur Recherche und Nutzung angeboten werden. Um eine einheitliche Nutzung zu ermöglichen, muss dabei ein über alle Marktplätze einheitliches Klassifikationsschema verwendet werden (siehe Abschnitt 3.3) und die einzelnen Komponenten müssen auf eine einheitliche Art und Weise spezifiziert werden (siehe Abschnitt 3.4).

Desweiteren ist es notwendig, dass eine Integration der Marktplätze in gängige Entwicklungsumgebungen erfolgt, so dass die Entwickler direkt aus der Entwicklungsumgebung nach passenden Komponenten recherchieren können. Diese Einbindung ist beispielsweise beim Marktplatz CompoNex realisiert. Die in diesem Marktplatz enthaltenen Komponenten (hauptsächlich auf Microsoft-Basis) können mittels der Web-Service-Technologie über ein PlugIn direkt in Microsofts Entwicklungsumgebung Visual Studio .NET verwendet werden (siehe auch [Over2002]).

Im Bereich der Marktplätze für Software-Komponenten sind durchaus auch branchen-, technologie- oder anwendungsgebietsspezifische Marktplätze sinnvoll, die sich auf ein gewisses Themengebiet fokussieren.

Außerdem sollte den Nutzern die Möglichkeit gegeben werden, analog zu einem „Schwarzen Brett“, Suchanzeigen nach den gewünschten Komponenten aufgeben zu können. Diese Anzeigen können durch die Komponentenhersteller durchsucht werden und diese können geeignet darauf reagieren, beispielsweise durch einen Hinweis auf eine Komponente in ihrem Portfolie oder durch ein Angebot zur Erstellung der gewünschten Komponente.

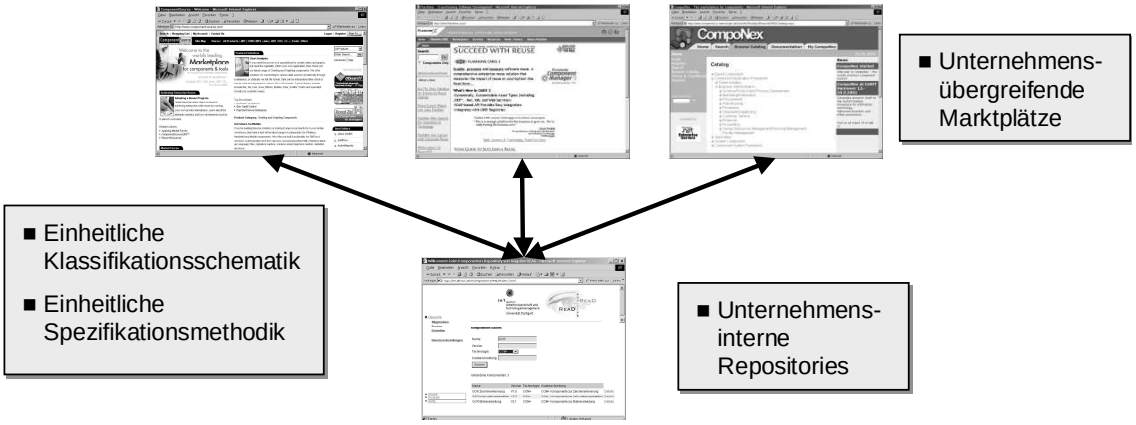


Abb. 4: Marktplätze und Repositories

Desweiteren ist die Einführung von unternehmensinternen Komponenten-Repositories notwendig, in denen die bereits im Unternehmen entwickelten und eingesetzten Komponenten abgelegt, verwaltet und recherchiert werden können. Dabei ist eine Anbindung und ein Abgleich mit den unternehmensübergreifenden Marktplätzen wünschenswert.

Auch bei den unternehmensinternen Repositories besteht die Forderung nach einer einheitlichen Klassifikation und Spezifikation der Komponenten. Neben den Basisdiensten für das Einstellen und Suchen von Komponenten sind an dieser Stelle durchaus auch Mehrwertdienste, wie z.B. Benachrichtigungsmechanismen, eine Unterstützung bei der Ermittlung von Kennzahlen im Bereich der Softwareentwicklung sowie die Implementierung von Anreizsystemen, denkbar. Im Rahmen des durch das BMBF geförderten Projekts Adaptive READ[7] wurde ein Prototyp eines derartigen Repositories entwickelt, das die beschriebenen Mehrwertdienste realisiert (siehe auch [HöWe2002a, WeHS2001]).

[7] www.adaptive-read.de

3.3 Einwicklung eines Klassifikationsschemas für Software-Komponenten

Wie in Abschnitt 3.2 beschrieben, ist die Entwicklung eines einheitlichen Klassifikationsschemas für Software-Komponenten eine notwendige Voraussetzung für die effiziente Nutzung von Marktplätzen und Repositories für Software-Komponenten.

Im Bereich der Sachgüter sind derzeit unterschiedliche Standardisierungsbemühungen im Gange. Beispiele hierfür sind der eher breit angelegte eCl@ss-Standard[8] und der auf die Baubranche spezialisierte Standard proficl@ss[9]. Beide kostenlos erhältlichen Standards definieren unterschiedliche Kategorien von Produkten, die hierarchisch angeordnet sind und jeweils unterschiedliche Merkmale besitzen, die wiederum jeweils unterschiedliche Wertebereiche beinhalten.

Diese Vorgehensweise gilt es nun auf Software-Komponenten zu übertragen. Auch auf diesem Gebiet gibt es unterschiedliche Klassifikationsansätze, die sich jedoch bisher noch nicht auf breiter Front durchgesetzt haben, so dass bisher jeder Marktplatz seine eigene Klassifikationsmethodik einsetzt.

Die oben genannten Standards im Sachgüterbereich beziehen sich auf sog. B- und C-Güter, d.h. Güter, deren Funktions- und Leistungsumfang relativ einfach zu standardisieren ist. A-Güter, d.h. Produkte und Dienstleistungen, die erst aufgrund von konkreten Kundenanforderungen produziert werden, sind auch dort bisher nicht standardisiert. Da Software-Komponenten prinzipiell auch eher in diese Kategorie fallen, ist es auch dort sehr schwierig zu einem einheitlichen Klassifikationsschema zu kommen.

[8] www.eclass.de

[9] www.proficlass.org

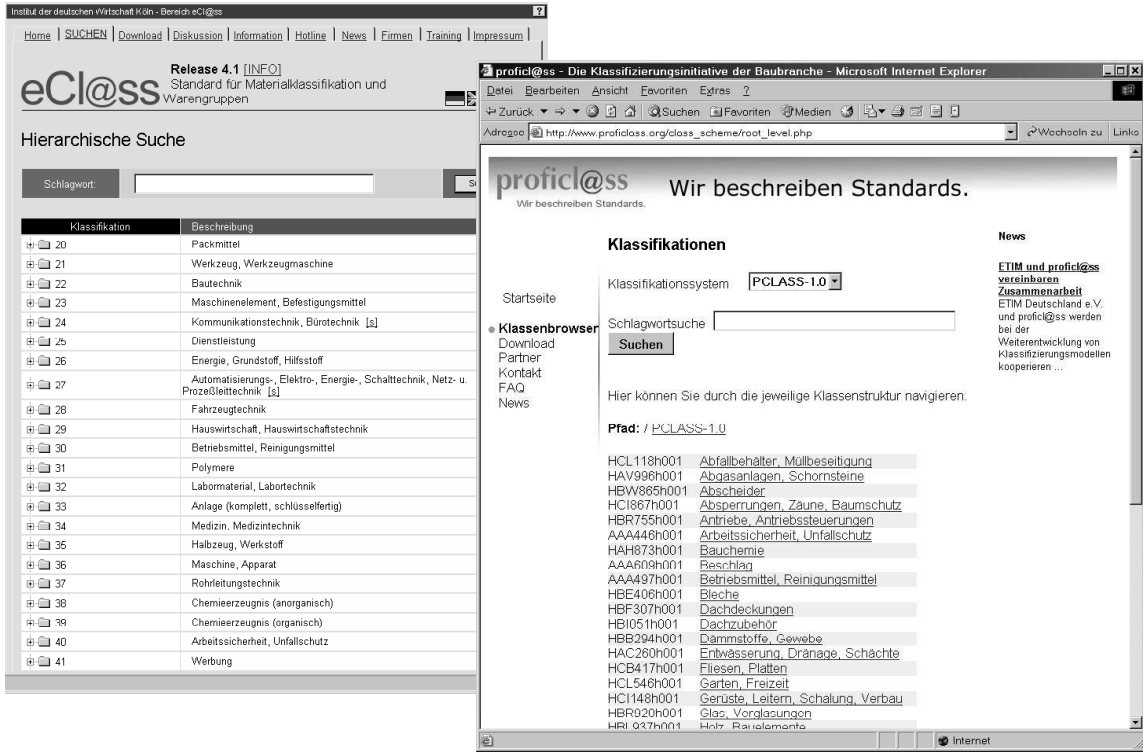


Abb. 5: Klassifikationsstandards im Sachgüterbereich

Ein erster Ansatz ist jedoch, die wesentlichen Merkmale, wie z.B. die verwendete Technologie, das Lizenzmodell, Angaben zum Hersteller usw., zu standardisieren. Die oberste Ebene einer möglichen Struktur einer Komponenten-Klassifikation ist in Abb. 6 dargestellt. Es werden identifizierende Merkmale, charakterisierende Merkmale, die benötigten Ressourcen, domänen-spezifische Merkmale, framework-spezifische Merkmale sowie komponenten-spezifische Merkmale definiert. Als technisches Medium der Beschreibung bieten sich an dieser Stelle XML-Technologien an.

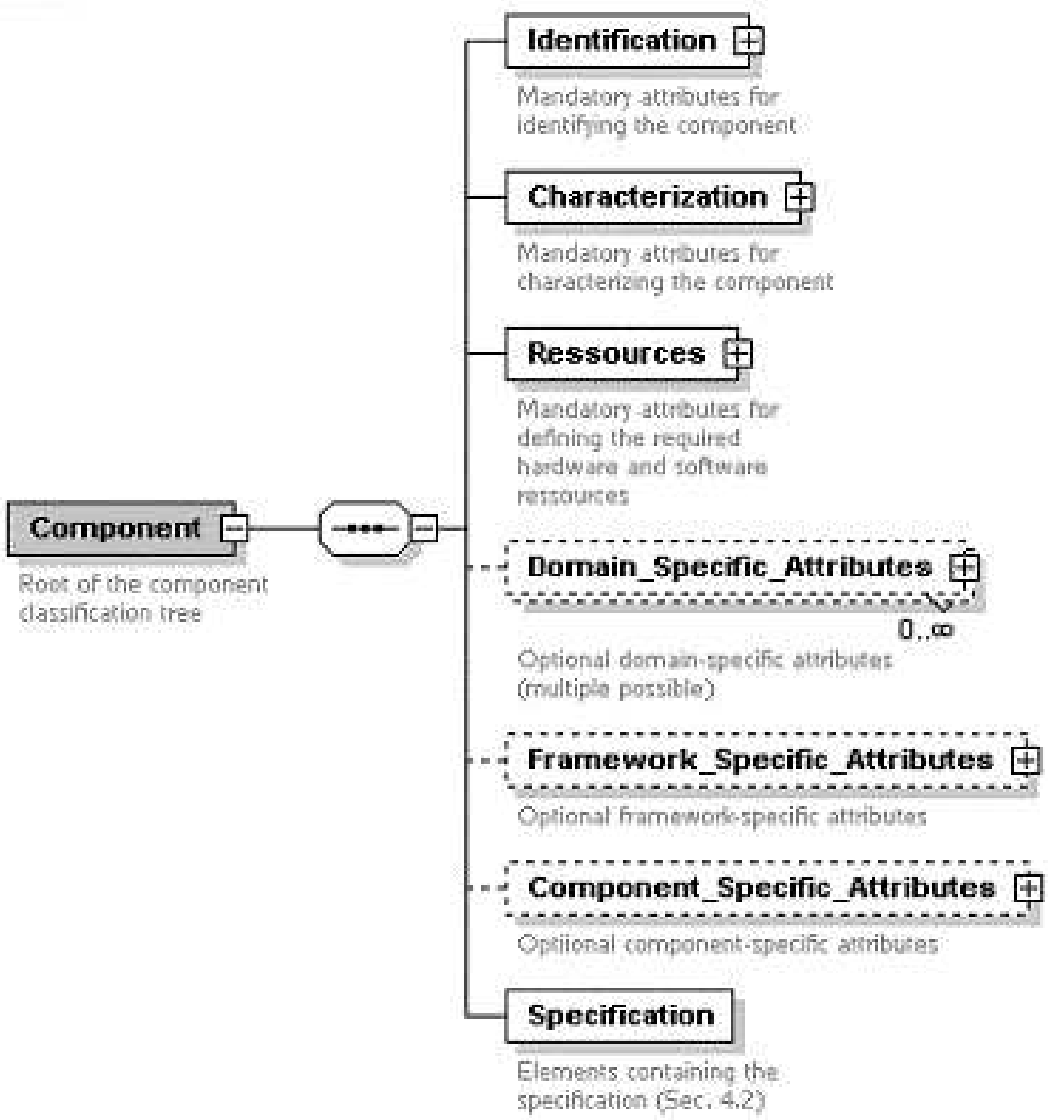


Abb. 6: Mögliche Struktur einer Komponenten-Klassifikation

3.4 Entwicklung einer einheitlichen Spezifikationsmethodik für Software-Komponenten

Während ein Klassifikationsschema hauptsächlich notwendig ist, um einzelne Komponenten aufzufinden, ist die Spezifikation notwendig, um die Funktionalität der Komponente zu verstehen. Dies ist insbesondere bei der Evaluation der Komponente und bei einem geplanten Einsatz notwendig (siehe Abb. 7).

Ein weiteres wichtiges Einsatzgebiet einer Spezifikation ist auch die Vergabe der Komponentenentwicklung an externe Zulieferer, z.B. auch ins Ausland. Nur wenn die

Funktionalität der Komponente ausreichend genau spezifiziert ist, kann der externe Partner die Funktionalität der Komponente zufriedenstellen implementieren. Auch eine anschließende Verifikation der Funktionalität der Komponente ist nur aufgrund dieser Spezifikation möglich.

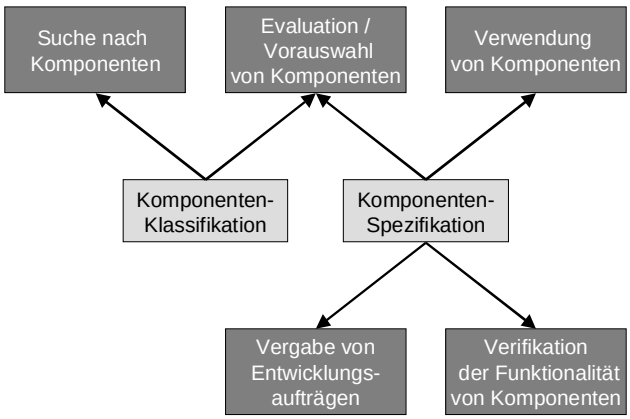


Abb. 7: Zusammenhang Klassifikation und Spezifikation

Auch im Bereich der Spezifikation von Komponenten gibt es mehrere Ansätze, aber noch keinen weithin akzeptierten Standard. Viele Versuche, Komponenten formal zu spezifizieren, scheitern an der darin inhärent vorhandenen Komplexität und somit an der mangelhaften Anwendbarkeit in der Praxis. Ein vielversprechender Ansatz ist die Nutzung der Unified Modeling Language (UML), eines Defacto-Standards für die Beschreibung von Software-Systemen. Da die UML schon weit verbreitet ist, besitzt sie auch eine dementsprechend gute Werkzeugunterstützung.

Einen ähnlichen Weg hat auch der Arbeitskreis 5.10.3 „Komponentenorientierte betriebliche Anwendungssysteme“ der Gesellschaft für Informatik beschritten und eine Spezifikationsmethodik für Fachkomponenten definiert, die auf Industriestandards, wie z.B. UML, aufsetzt und diese, wo notwendig, um zusätzliche Beschreibungsmittel ergänzt. Eine Beschreibung dieser Spezifikationsmethodik ist im Internet verfügbar [ABC+2002].

Eine Übersicht über die darin enthaltenen Spezifikationsebenen, die jeweils unterschiedliche Aspekte abdecken, ist in Abb. 8 dargestellt.

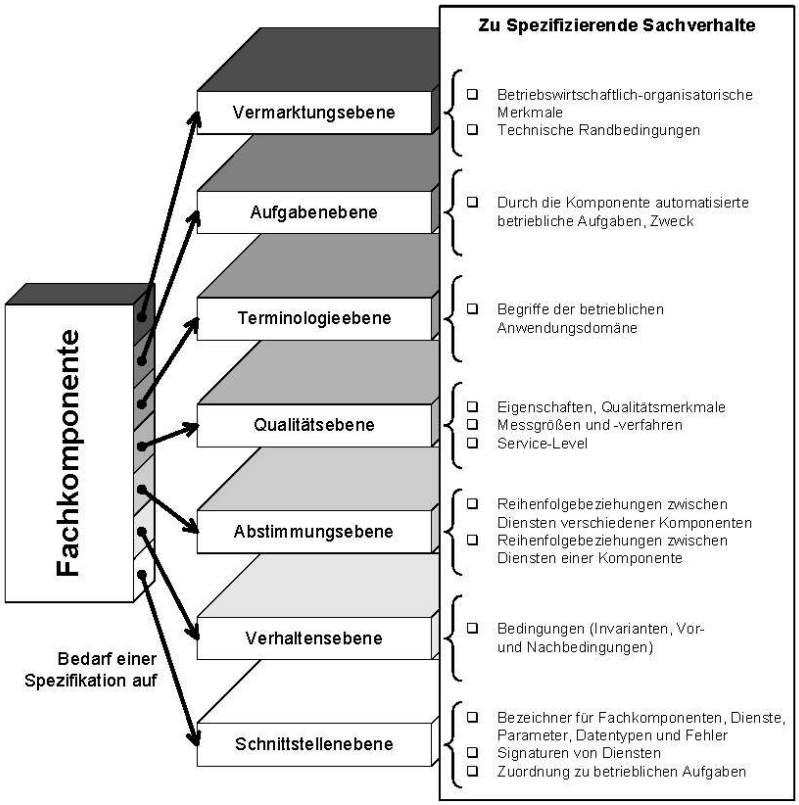


Abb. 8: Spezifikationsmethodik des GI-Arbeitskreises 5.10.3 [ABC+2002]

3.5 Entwicklung von unternehmensübergreifenden Fachkonzepten

In der in Abb. 8 im Überblick dargestellten Spezifikationsmethode werden auf der „Terminologieebene“ die Begriffe der betrieblichen Anwendungsdomäne referenziert. Eine Standardisierung dieser Begriffe und der dahinterliegenden Fachkonzepte ist eine notwendige Vorraussetzung für eine unternehmensübergreifende Wiederverwendung von Software-Komponenten. Ansonsten harmonieren unterschiedliche Komponenten zwar auf technischer Ebene, die dahinterliegenden Fachkonzepte sind jedoch evtl. unterschiedlich und verhindern eine „Kompatibilität“ auf fachlicher Ebene, die durch technische Maßnahmen i.d.R. nicht zu überbrücken ist. Ein Beispiel hierfür sind Unterschiedliche Konzepte eines Kunden oder eines Auftrags.

Es ist zweckmäßig, eine derartige Standardisierung zuerst innerhalb einer Branche durchzuführen, da es selbst innerhalb einer Branche nicht trivial ist, die Interessen der unterschiedlichen Unternehmen zu vereinheitlichen.

Beispielhaft ist in dieser Beziehung die Versicherungsbranche. Der Gesamtverband der deutschen Versicherungswirtschaft hat mit der VAA-Architektur (Anwendungsarchitektur der deutschen Versicherungswirtschaft) ein Fachkonzept in Form eines objektorientierten Modells geschaffen, das Komponentenhersteller als Grundlage nutzen können, um Software-Komponenten für diese Branche herzustellen. Durch das einheitliche Fachkonzept wird eine Austauschbarkeit der Komponenten über Unternehmensgrenzen ermöglicht [GdV2001]. Ein beispielhafter Ausschnitt aus dem fachlichen Modell ist in Abb. 9 dargestellt.

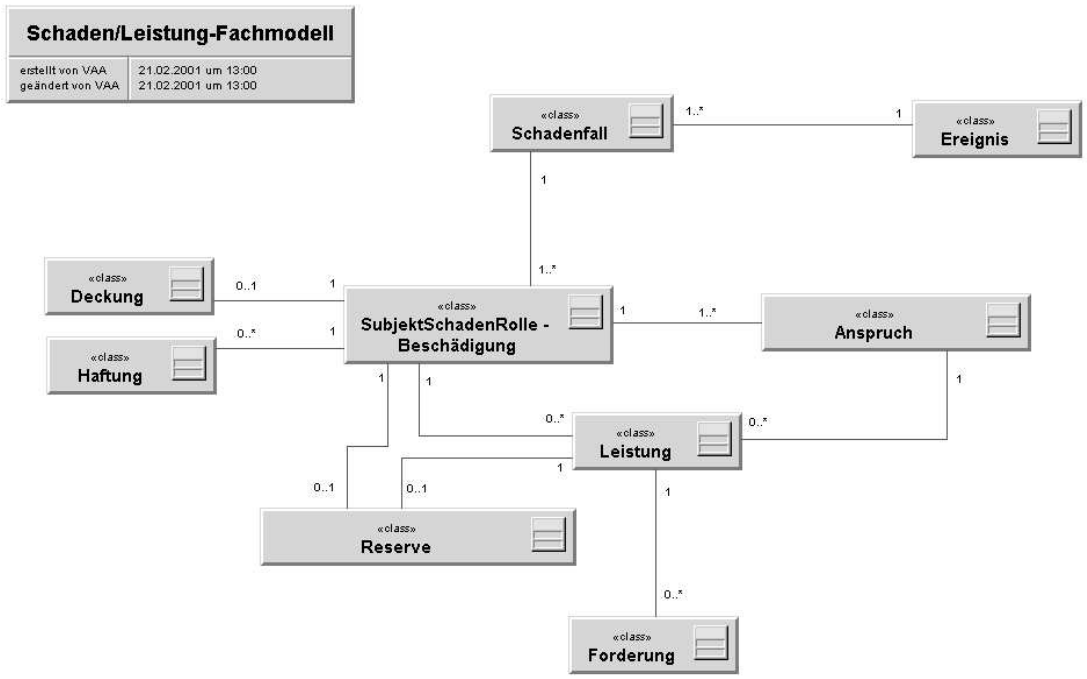


Abb. 9: Ausschnitt aus dem fachlichen Modell der Versicherungsbranche

Der Ansatz, ein fachliches Modell eines gesamten Anwendungsgebiets bzw. einer gesamten Anwendungsdomäne zu erstellen, ist auch als „Domain Analysis“ bzw. „Domain Engineering” bekannt. In [Sodhi1998] wird eine derartige Domänenanalyse als notwendige Voraussetzung für eine unternehmensübergreifende Wiederverwendung beschrieben. Insbesondere besitzt diese Vorgehensweise einen Vorteil, wenn man Software-Produktlinien, d.h. mehrere ähnliche Systeme innerhalb derselben Anwendungsdomäne, erstellt (siehe auch [ABF+2000]).

3.6 Einführung der Wiederverwendung als Teil der Unternehmensstrategie

Eine systematische, an der produzierenden Industrie orientierte, Vorgehensweise bei der industriellen Herstellung von Software hat auch Auswirkungen auf die Organisationsstruktur der Unternehmen sowie das auf gewünschte Verhalten der Mitarbeiter und des Managements.

Eine wesentliche organisatorische Veränderung ist die Einführung einer Organisationseinheit, die für die Wiederverwendung von Software-Komponenten zuständig ist. Folgende Tätigkeiten sind Teil des Aufgabenspektrums dieses „Wiederverwendungsteams“:

- Untersuchung der externen Komponentenmärkte nach wiederverwendbaren Komponenten
- Evaluation und Aquisition von Komponenten
- Pflege der Kontakte zu Komponenten-Zulieferern
- Coaching der Projekte hinsichtlich Wiederverwendung von bestehenden Komponenten und der Verwendung von geeigneten Architekturen
- Betrieb und Pflege eines internen Repositories
- Ermittlung von Kennzahlen bzgl. der Software-Entwicklungsprojekte
- ...

Die ersten drei Tätigkeitsfelder entsprechen weitestgehend dem Profil der Einkaufsabteilung in produzierenden Unternehmen. Wichtig ist auch die Ermittlung von Kennzahlen, um dem Management eine Grundlage für die Entscheidung für oder gegen den Einsatz von extern entwickelten Komponenten zu geben.

Ein wichtiger Faktor ist auch der einzelne Mitarbeiter bzw. Entwickler. In Abschnitt 2.3 wurde dargestellt, dass manche Entwickler vorhandene Komponenten auch dann nicht wiederverwenden, wenn diese vorhanden und bekannt sind. Es ist daher notwendig, vor der Eigenentwicklung jeder Teilkomponente eines komplexen Systems eine obligatorische, explizit zu dokumentierende Make-Or-Buy-Entscheidung aufgrund von genau definierten Kriterien durchzuführen, um ein Kostenbewusstsein bei den

Mitarbeitern zu wecken und eine Wiederverwendung von bestehenden Komponenten zu motivieren.

Gegebenenfalls sind auch Anreizsysteme einzusetzen, die die Entwickler von wiederverwendbaren Komponenten, aber auch die Nutzer von Komponenten entsprechend belohnen (siehe auch [HöWe2002a]).

3.7 Klärung der rechtlichen Grundlagen

Ein wesentlicher Faktor ist auch die Schaffung von verbindlichen rechtlichen Grundlagen, die die Kooperation von unterschiedlichen Teilnehmern des Software-Marktes regeln. Dabei ist ein besonderer Schwerpunkt auf die Anwendbarkeit der Regelungen zu legen. Das bedeutet, dass die jeweiligen Fragestellungen nicht nur aus juristischer, sondern auch aus software-technischer Sicht betrachtet werden müssen, was zu einem Wachstum der Disziplin der Rechtsinformatik führen muss.

4. Zusammenfassung und Ausblick

Der Beitrag hat ausgehend von den Hemmnissen, die die Kooperation von unterschiedlichen Marktteilnehmern an einem Software-Komponenten-Markt behindern, unterschiedliche Lösungsansätze vorgestellt, um dem Ideal einer Software-Komponenten-Industrie näher zu kommen.

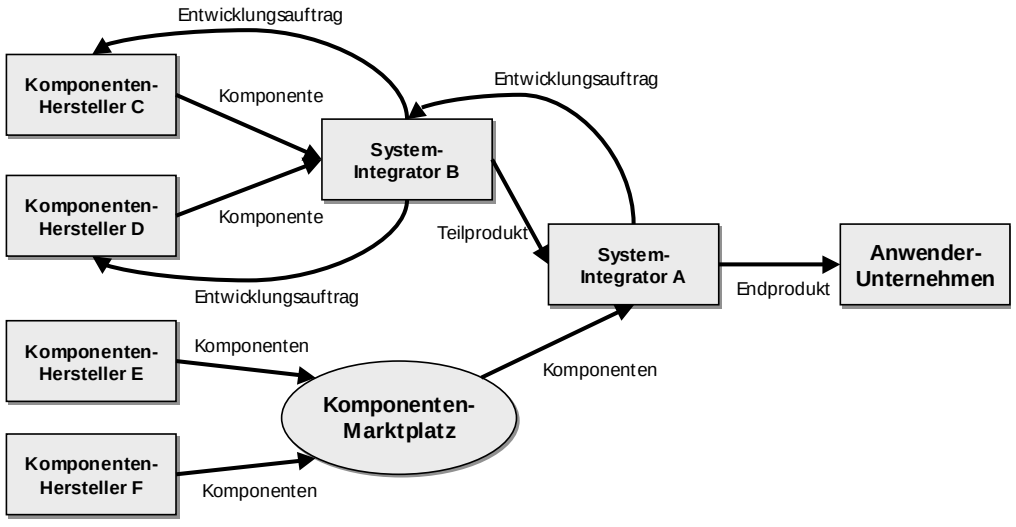


Abb. 10: Ein Software-Komponenten-Markt

Eine derartige Software-Komponenten-Industrie ist geprägt von Komponenten-Herstellern, System-Integratoren und Anwendern, die über Marktplätze oder direkt miteinander interagieren. In Abb. 1010 ist eine beispielhafte Kooperationsstruktur dargestellt. Der System-Integrator A liefert ein Software-Produkt an Anwender-unternehmen. Dazu bedient er sich eines Teilprodukts, das er bei System-Integrator B in Auftrag gegeben hat, der wiederum die Komponenten-Hersteller C und D mit der Herstellung von Teilkomponenten beauftragt. Desweiteren benutzt der System-Integrator A weitere Komponenten, die er von einem Komponenten-Marktplatz bezogen hat. Diese Komponenten wurden proaktiv, d.h. ohne konkreten Auftrag und ohne Kenntnis der späteren Verwendung, von den Komponenten-Herstellern E und F produziert und auf dem Marktplatz eingestellt.

Die Reife eines derartigen Software-Marktes kann mit betriebswirtschaftlichen Methoden ermittelt werden [Hahn2002]. Derzeit ist der Markt in Deutschland, aber auch weltweit, noch auf einer relativ niedrigen Reifestufe. Ein Land, eine Region oder ein Unternehmen, das jedoch auch in Zukunft wettbewerbsfähig sein möchte, muss Anstrengungen unternehmen, die Potenziale zu realisieren, die durch eine efiziente Entwicklung und einen effizienten Einsatz von Software erzielt werden können.

Das Fraunhofer IAO unterstützt Unternehmen dabei, diesen Herausforderungen zu begegnen, indem die neuesten Erkenntnisse aus aktuellen Forschungsprojekten gemeinsam mit den jeweiligen Unternehmen in die Praxis umgesetzt werden.

5. Literatur

[ABC+2002] Ackermann, J.; Brinkop, F.; Conrad, S.; Fettke, P.; Frick, A.; Glistau, E.; Jaekel, H.; Kotlar, O.; Loos, P.; Mrech, H.; Ortner, E.; Raape, U.; Overhage, S.; Sahm, S.; Schmietendorf, A.; Teschke, T; Turowski, K.: Vereinheitlichte Spezifikation von Fachkomponenten. Memorandum des GI-Arbeitskreises 5.10.3: Komponentenorientierte betriebliche Anwendungssysteme, Februar 2002. http://www.fachkomponenten.de

[ABF+2000] Anastasopoulos, M.; Bayer, J.; Flege, O.; Gacek, C.: A Process for Product Line Architecture Creation and Evaluation - PuLSE-DSSA Version 2.0. IESE Report 038.00/E, Fraunhofer IESE. Kaiserslautern, 2000.

[AlFr1998] Allen, P.; Frost, S.: Component-Based Development for Enterprise Systems - Applying the Select Perspective. Cambridge University Press, Cambridge 1998.

[DiEs2001] Dietzsch, A.; Esswein, W.: Gibt es eine „Softwarekomponenten-Industrie“ ? Ergebnisse einer empirischen Untersuchung. In: Buhl, H. U.; Huther, A.; Reitwiesner, B. (Hrsg.): Information Age Economy, 5. Internationale Tagung Wirtschaftsinformatik 2001, Tagungsband, 19.-21.9.2001, Augsburg, Physika-Verlag, ISBN 3-7908-1427-X, Heidelberg 2001, S. 697 – 710.

[DrWi1999] Dröschel, W.; Wiemers, M.: Das V-Modell 97 - Der Standard für die Entwicklung von IT-Systemen mit Anleitung für den Praxiseinsatz. Oldenbourg Wissenschaftsverlag, ISBN 3-486-25086-8, München 1999.

[DsWi1998] D’Souza, D. F.; Wills, A. .C.: Objects, Components and Frameworks with UML: The Catalysis Approach, Addison Wesley, Reading 1998.

[FeInLo2002] Fettke, P.; Intorsureanu, I.; Loos, P.: Komponentenorientierte Vorgehensmodelle im Vergleich. In: Turowski (Hrsg.): 4. Workshop komponentenorientierte betriebliche Anwendungssysteme (WKBA 4), Universität Augsburg und Gesellschaft für Informatik, Tagungsband, Augsburg 11.6.2002, S. 19-43.

[GdV2001] Gesamtverband der deutschen Versicherungswirtschaft e.V.: Die Anwendungsarchitektur der Versicherungswirtschaft. Final Edition. http://www.gdv-online.de/vaa/vaafe_html/index.htm

[GfK+2000] GfK Marktforschung GmbH; Fraunhofer-Institut für Experimentelles Software Engineering (IESE); Fraunhofer-Institut für Systemtechnik und Innovationsforschung (ISI): Analyse und Evaluation der Softwareentwicklung in Deutschland. Abschlussbericht einer Studie für das Bundesministerium für Bildung und Forschung, Dezember 2000. http://www.dlr.de/IT/IV/Studien/evasoft_abschlussbericht.pdf

[Hahn2002] Hahn, H.: Ein Modell zur Ermittlung der Reife des Softwaremarktes. In: Turowski (Hrsg.): 4. Workshop komponentenorientierte betriebliche Anwendungssysteme (WKBA 4), Universität Augsburg und Gesellschaft für Informatik, Tagungsband, Augsburg 11.6.2002, S. 57-74.

[HöWe2002a] Höß, O.; Weisbecker, A.: Konzeption eines Repositories zur Unterstützung der Wiederverwendung von Softwarekomponenten. In: Turowski (Hrsg.): 4. Workshop komponentenorientierte betriebliche Anwendungssysteme (WKBA 4), Universität Augsburg und Gesellschaft für Informatik, Tagungsband, Augsburg 11.6.2002, S. 57-74.

[HöWe2002b] Höß, O.; Weisbecker, A.: Componentware verspricht Erfolg im E-Business. Computerwoche CeBIT-Sonderausgabe 2002, S. 50-51.

[HöWe2001] Höß, O.; Weisbecker, A.: Komponentenbasierte Software für Produkte und Dienstleistungen (KoSPuD): Ergebnisse und Erfahrungen eines praxisorientierten Verbundforschungsprojekts. In: Turowski, K. (Hrsg.): 3. Workshop komponentenorientierte betriebliche Anwendungssysteme (WKBA 3), Universität der Bundeswehr München und Gesellschaft für Informatik, Tagungsband, Frankfurt 4.4.2001, S. 1-13.

[Kruc1999] Kruchten, P.: Der Rational Unified Process – Eine Einführung, Addison-Wesley, 1999.

[Over2002] Overhage, S.: Die Spezifikation – kritischer Erfolgsfaktor der Komponentenorientierung. In: Turowski (Hrsg.): 4. Workshop komponentenorientierte betriebliche Anwendungssysteme (WKBA 4), Universität Augsburg und Gesellschaft für Informatik, Tagungsband, Augsburg 11.6.2002, S. 1-17.

[Sodhi1998] Sodhi, J.; Sodhi, P.: Software Reuse – Domain Analysis and Design Processes, McGraw-Hill, New York 1998.

[WeHS2001] Weisbecker, A.; Höß, O.; Strauß, O.: Organisatorische und technische Maßnahmen zur Wiederverwendung von Komponenten. Deliverable D2.7, Projekt Adaptive READ, gefördert durch das BMBF, Stuttgart 2001.

[Weis2000] Weisbecker, A. (Hrsg.): KoSPuD – Komponentenbasierte Software für Produkte und Dienstleistungen. Abschlussbericht, Fraunhofer IRB Verlag, ISBN 3-8167-5556-9, Stuttgart 2000.